Анализ поведения дискретных динамических систем средствами логического программирования [1]

М.И.Дехтярь, А.Я.Диковский

Аннотация

Работа содержит определения некоторых свойств устойчивого поведения дискретных динамических систем (в частности, - стабильности и гомеостатичности). Определения формулируются в терминах логических программ с модификациями над реляционными базами данных. Формальная модель, используемая нами, основана на новом понятии - понятии ограниченного воздействия внешней среды. Обсуждается возможность создания специфической инструментальной среды для анализа и моделирования поведения динамических систем на основе исследуемых понятий.

1 Введение

Эта работа развивает подход к анализу поведения дискретных динамических систем средствами логического программирования, предложенный в наших предыдущих публикациях [2, 3, 4].

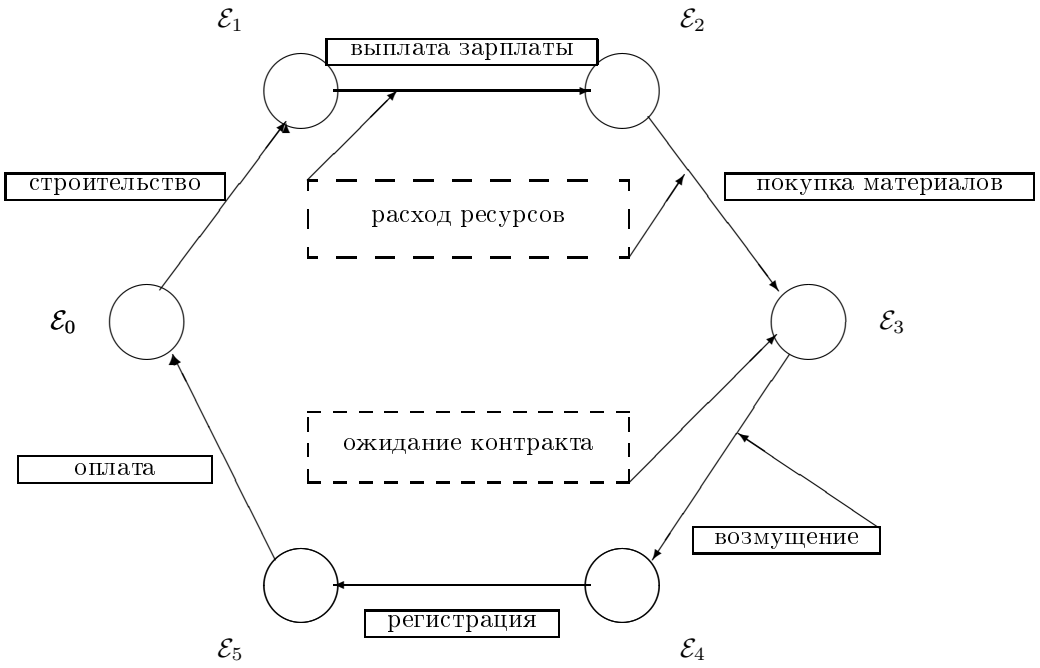
Языки логического программирования (ЛП) являются гибким средством интенсионального определения данных. Они значительно обобщают средства определения данных, доступные в языках запросов к реляционным базам данных (последние, как известно [14], эквивалентны нерекурсивному фрагменту языка Дейталог, расширенному ограниченным отрицанием). В большинстве языков ЛП так или иначе различаются экстенсиональная и интенсиональная части программы. Первая, как правило, содержит конечный набор фактов (*состояние БД*), доступный для динамических изменений, вторая аксиоматически определяет потенциально бесконечное множество интенсиональных фактов, вытекающих из фактов этого состояния. Как правило, динамические изменения состояний БД задаются с помощью операторов модификации, добавляющих и удаляющих факты. Например, в Дейталоге и Прологе это - динамические предикаты $assert/1, retract/1$, в LDL [11] это динамические операторы $+, -$, в дедуктивных базах данных это - операторы элементарных транзакций $insert, delete$. В литературе нет однозначного толкования термина "дедуктивная база данных"(ДБД), однако под этим чаще всего понимается обобщенная (т.е. расширенная некоторым вариантом отрицания) логическая программа, дополненная некоторым набором условий (*ограничений целостности*), которым должны удовлетворять динамически изменяющиеся состояния БД [7]. ДБД являются удобным математическим аппаратом описания поведения дискретных динамических систем. В самом деле, состояние системы естественно представляется в виде экстенсионального состояния БД. Законы поведения системы, описывающие правила изменения ее состояний, задаются интенсиональными аксиомами, т.е. предложениями логической программы. Критерии допустимости состояний могут формулироваться в виде формул-ограничений целостности.

Для изучения поведения динамических систем традиционно используется аналитический аппарат, который по отношению к дискретным системам обладает по крайней мере двумя серьезными недостатками: во-первых, его применение основано на предположении о непрерывности изменения состояний (близости соседних состояний), которое редко непосредственно выполняется для дискретных систем, во-вторых, аналитический аппарат весьма чувствителен к большому числу параметров дискретных задач. Для описания же законов поведения дискретных систем в терминах ДБД не требуется никаких предположений о непрерывности, да и число параметров задачи является второстепенным фактором.

[1] Эта работа была поддержана грантом INTAS (Grant 94-2412) и Российским Фондом фундаментальных исследований (грант 95-01-01321).

В теории дедуктивных баз данных модификации, нарушающие ограничения целостности IC, как правило рассматриваются как недопустимые (см. [7, 9, 12]). В этом состоит одно из главных отличий предлагаемого нами подхода от традиционного. Мы ориентируемся на важный класс приложений ДБД, в которых они рассматриваются как модели дискретных динамических систем, функционирующих в **активной внешней среде.** При этом их поведение (равно как и реакции среды) **зависит от некоторых ресурсов.** В этих приложениях объемы доступных ресурсов ограничены, и соответствующие ограничения могут быть заданы как предложениями логической программы, так и с помощью ограничений целостности. Возможные реакции внешней среды на действия ДБД представляются как непосредственные изменения состояний БД. И те, и другие могут влиять на объем доступных ресурсов. Отсюда возникает основное отличие в подходах к трактовке ограничений целостности. Мы различаем два источника изменений состояний БД: *модификации,* вызванные действиями самой ДБД, и внешние *возмущения* среды, в которой эта ДБД действует. В ходе взаимодействия этих двух сторон воздействия одной стороны могут вызывать нарушения ограничений целостности IC, а другой - их восстановление. Рассмотрим следующий содержательный пример.

Предположим, что некоторая фирма заключает контракты и занимается строительством дачных домиков. Финансирование ее деятельности полностью зависит от платежей заказчиков. Полученные от них суммы зачисляются на текущий счет и тратятся на материалы и на выплату ежемесячной зарплаты служащим (см. Рис. 1). В состоянии $\mathcal{E}_0$ фирма располагает контрактом и строит дом. В следующем состоянии $\mathcal{E}_1$ она платит зарплату сотрудникам. Таким образом, переход в состояние $\mathcal{E}_2$ связан с расходом некоторых ресурсов. Будучи в состоянии $\mathcal{E}_2$, фирма приобретает необходимые материалы, снова затрачивая финансовые ресурсы, и оказывается в недопустимом состоянии $\mathcal{E}_3$. В этом состоянии финансовые ресурсы исчерпаны и невозможно совершить какое-либо действие до тех пор, пока не будет заключен и оплачен новый контракт. Таким образом, $\mathcal{E}_3$ - это состояние ожидания контракта на строительном рынке. Мы подчеркиваем, что появление контракта на рынке **не регулируется правилами функционирования фирмы.** Это событие является **внешним** по отношению к деятельности фирмы и представляет "возмущение"среды, в которой она действует. После появления требуемого контракта, переводящего систему в допустимое состояние $\mathcal{E}_4$, он регистрируется (переход в состояние $\mathcal{E}_5$), деньги помещаются на счет и система возвращается в начальное состояние $\mathcal{E}_0$. Следовательно, фирма может продолжать свою деятельность при условии, что на рынке будет иметься достаточное число контрактов.



.1

Приведенный пример показывает, что в такого рода приложениях центральной задачей является поиск стратегии поведения, гарантирующей теоретически бесконечный процесс жизнедеятельности системы - *устойчивое* ее поведение. Действия системы и возмущения среды чередуются во времени, и таким образом, возможное поведение системы в активной среде описывается траекториями в пространстве состояний, в которых переходы из состояния в состояние попеременно определяются то некоторым действием системы, то некоторым возмущением среды. Предположим для определенности, что процесс взаимодействия между системой и внешней средой всегда **начинается в допустимом состоянии,** и что первый шаг всегда выполняется той стороной, **которая может нарушить IC.** В целях достижения устойчивости другая сторона должна стараться **восстановить IC.** Таким образом, мы приходим к двум дуальным режимам устойчивого поведения. Режим, в котором первый шаг делается системой (как в нашем примере), мы называем *стабильным.* Стабильность означает, что нарушения IC, вызванные действиями динамической системы, могут быть исправлены за счет возмущений (коррекций) ее внешней среды. Иначе говоря, система может функционировать бесконечно **за счет этих возмущений.**

Двойственный режим устойчивого поведения, при котором первый шаг выполняет среда, мы называем *гомеостатичным.* Гомеостатичность означает, что нарушения IC, вызванные возмущениями внешней среды, могуть быть исправлены за счет действий динамической системы. Другими словами, несмотря на разрушительные возмущения среды, система может функционировать бесконечно. Эти понятия, конечно, требуют уточнений, без которых всегда можно восстановить выполнение IC для обеих моделей, по крайней мере теоретически. Для этого после всякого нарушения IC противоположной стороне достаточно просто восстанавливать предыдущее допустимое состояние. Разумеется, в приложениях с ограниченными ресурсами такое восстановление возможно не всегда. Поэтому разумно ввести некоторую меру объема множеств экстенсиональных фактов. Простейшим способом измерения объема возмущений и состояний является теоретико-множественный: состояния и верхние границы возмущений являются конечными множествами экстенсиональных атомов, а порядок на них задается отноше-

нием включения на множествах. При этом мы считаем траекторию $\{\delta\}-$*ограниченной*, если в ней размер каждого возмущения не превосходит конечного множества δ (точнее, не превосходит с точностью до информационной эквивалентности). Помимо этого мы рассматриваем также и другие ограничения на возмущения μ, более общие, но тоже быстро распознаваемые. Итак, мы рассматриваем $\mu-$ограниченные стабильные и гомеостатичные траектории. В результате мы получаем различные релятивизованные модели устойчивого поведения. Например, $\mu-$ограниченная $\exists\exists-$стабильность в заданном состоянии (проиллюстрированная первым примером) означает непустоту множества бесконечных $\mu-$ограниченных стабильных траекторий, начинающихся в этом состоянии. Ниже будут приведены и другие примеры устойчивого поведения, например, $\mu-$ограниченного $\forall\exists-$гомеостатичного поведения в данном состоянии, которое состоит в том, что для каждого $\mu-$ограниченного возмущения, изменяющего данное состояние, существует некоторая модификация, приводящая к состоянию, обладающему тем же свойством. Другими словами, это означает, что система может исправлять все $\mu-$ограниченные деструктивные возмущения. В приложениях встречаются и некоторые другие виды неограниченно устойчивого поведения (например, $\mu-$ограниченная $\forall\exists-$стабильность или $\mu-$ограниченная $\forall\forall-$гомеостатичность).

Понятия $\mu-$ограниченной стабильности и гомеостатичности характеризуют устойчивое поведение динамических систем на допустимых состояниях. Мы будем также рассматривать свойство $\mu-$ограниченной *перспективности*, которое характеризует поведение системы в недопустимых состояниях. Данное состояние $\mathcal{E}$ перспективно, если из него можно достичь посредством конечной $\mu-$ограниченной траектории некоторое допустимое состояние. Таким образом, пространство состояний разбивается на два класса: состояния из которых можно достичь допустимое состояние - $\mu-$*перспективные* состояния, и те, из которых нельзя - $\mu-$*неперспективные* состояния. На самом деле, свойство $\mu-$ограниченной перспективности, которое рассматривается ниже, является $\mu-$ограниченной $\exists\exists-$перспективностью. Как и для стабильности, можно определить естественным образом и другие виды $\mu-$ограниченной перспективности, используя различные комбинации кванторов.

В этой работе, перечисленные здесь во введении свойства устойчивого поведения дискретных систем, получат точные определения и будут проиллюстрированы разнообразными примерами. Мы обсудим сложность этих свойств, и на базе проведенного анализа обрисуем основные черты инструментальной среды для экспериментов с дискретными динамическими системами и для анализа их поведения.

2 Исходные понятия

Мы рассматриваем различные варианты языков логического программирования в сигнатуре *экстенсиональных* предикатов $\mathbf{P}^e$ и (возможно пустой) сигнатуре *интенсиональных* предикатов $\mathbf{P}^i$. Эти сигнатуры не пересекаются. Экстенсиональные предикаты обозначают факты, описывающие состояния системы, интенсиональные предикаты обозначают свойства состояний, косвенно определяющиеся через эти факты. Через $\mathbf{H}$ мы обозначаем множество всех *базисных* термов, т.е. термов построенных только из символов констант и функторов (сигнатура которых также зафиксирована). Через $\mathbf{B}^e$ мы обозначаем множество всех экстенсиональных атомов (*базисных фактов*), т.е. выражений вида $p(t_1,...,t_k)$, в которых $p \in \mathbf{P}^e$, и $t_1,...,t_k \in \mathbf{H}$. Состояния дискретных систем - это конечные подмножества $\mathbf{B}^e$. Мы будем определять воздействие дискретных систем на их состояния в терминах бинарных отношений $\vdash$, определенных на множестве состояний. В конечном счете эти отношения будут определяться через *элементарные модификации* состояний: $insert(p(u_1,...,u_k))$ - добавление к состоянию факта $p(u_1,...,u_k)$, и $delete(q(v_1,...,v_l))$ - удаление из состояния факта $q(v_1,...,v_l)$. Для экономии места мы допускаем и использование модификации *change(A,B)* вместо комбинации модификаций *delete(A), insert(B).*

В реальных приложениях данные в базах данных естественно разбиваются на классы, определяемые тем, что некоторые аргументы (атрибуты) отношений играют чисто информационную роль, т.е. их изменение не влияет на существенные свойства баз. Данные, различающиеся только такими "несущественными"чертами, можно считать эквивалентными. Такое разбиение данных на классы иногда позволяет уменьшить потенциально бесконечную необозримую прикладную область до конечной или более обозримой. Мы вводим для отражения этой ситуации следующее понятие.

Определение 1 *Пусть* $\equiv$ - *некоторая эквивалентность на* $\mathbf{H}$. *Ее можно естественным образом распространить на* $\mathbf{B}^e$: $g(t_1,...,t_n) \equiv g(t'_1,...,t'_n)$ *тогда и только тогда, когда* $t_i \equiv t'_i$ *для всех* $1 \leq i \leq n$. *Для любых* $D_1, D_2 \subseteq \mathbf{B}^e$ *мы полагаем* $D_1 \Leftarrow D_2$, *если* $D_1/_{\equiv} \subseteq D_2/_{\equiv}$. $D_1 \Leftrightarrow D_2$ *если* $D_1 \Leftarrow D_2$ *и* $D_2 \Leftarrow D_1$.

Отношение на состояниях $\vdash$ согласовано с эквивалентностью $\equiv$, *если для любых трех состояний* $\mathcal{E}_1, \mathcal{E}'_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$, *таких что* $\mathcal{E}_1 \Leftrightarrow \mathcal{E}_2$ *и* $\mathcal{E}_1 \vdash \mathcal{E}'_1$ *существует такое состояние* $\mathcal{E}'_2$, *что* $\mathcal{E}'_1 \Leftrightarrow \mathcal{E}'_2$ *и* $\mathcal{E}_2 \vdash \mathcal{E}'_2$.

Формула Φ согласована с эквивалентностью $\equiv$, *если для любых состояний* $\mathcal{E}_1, \mathcal{E}_2 \subseteq \mathbf{B}^e$ *из* $\mathcal{E}_1 \Leftrightarrow \mathcal{E}_2$ *следует, что* $\mathcal{E}_1 \models \Phi$ *эквивалентно* $\mathcal{E}_2 \models \Phi$.

Согласованность с информационной эквивалентностью является вполне разумным ограничением, соответствующим сути дела во многих приложениях. Скажем, в нашем примере со строительной фирмой база данных контрактов реализуется посредством предиката $contract(Number, Customer, Sum)$. При этом естественно считать эквивалентными все факты вида $contract(n_i, c_i, s_i)$, отличающиеся только именами заказчиков c_i, поскольку эти имена никак не влияют на финансовое состояние и поведение фирмы.

Мы будем считать, что эквивалентности $\equiv$ являются просто вычислимыми (например, за время пропорциональное размеру атомов). В большинстве рассматриваемых примеров эта эквивалентность является просто равенством.

Помимо приведенных здесь обозначений мы будем использовать и некоторые стандартные обозначения и понятия (например, понятие подстановки, частного случая, унификатора, и проч.) [8], а также - следующую классификацию логических программ с модификациями.

Определение 2 *Логическая программа* $\mathcal{P}$ *является* позитивной, *если в ней нет отрицаний. Она называется* базисной, *если все ее предложения базисны, т.е. не содержат переменных. Она является* плоской, *если все термы в ее предложениях являются переменными или константами. Мы назовем* $\mathcal{P}$ расширяющей, *если в ее предложениях не используется модификация delete/1.*

3 Продукционные динамические БД

Первый и наиболее простой язык для описания поведения дискретных динамических систем, который мы рассмотрим, строится на хорошо известном из области искусственного интеллекта понятии *продукции.* Как уже сказано, состояния дискретных систем представляются состояниями баз данных (БД), т.е. конечными множествами экстенсиональных фактов. При этом прямолинейном способе представления состояний системы столь же непосредственно описывается и элементарное преобразование состояния - при выполнении некоторого условия удалить некоторые факты и добавить некоторые другие. Такое преобразование описывается как *продукция,* т.е. предложение π вида

$$Con_1 \& \ldots \& Con_k \Rightarrow Act_1, \ldots, Act_m.$$

Каждое элементарное условие Con_i в этом предложении имеет вид α или $\neg\alpha$, где α - либо экстенсиональный атом, возможно содержащий предметные переменные, либо - встроенный (например, арифметический) предикат. Каждое действие Act_j является либо элементарной

модификацией $insert(p(u_1,...,u_k))$, либо элементарной модификацией $delete(q(v_1,...,v_l))$. Предполагается, что переменные, содержащиеся в аргументах элементарных модификаций или встроенных предикатов, входят в предшествующие *позитивные,* т.е. не содержащие отрицания, элементарные условия.

Содержательно, выполнение продукции заключается в проверке конъюнкции условий в ее левой части и в выполнении действий правой части в случае успеха этой проверки. Более точно, каждая продукция π задает отношение перехода $\vdash^\pi$ на множестве состояний, определяемое через стандартное понятие (базисной) подстановки: конечной функции, отображающей переменные в (базисные) термы. Отношение $\mathcal{E}_1 \vdash^\pi \mathcal{E}_2$ имеет место, если существует базисная подстановка σ, определенная в точности на переменных, входящих в позитивные условия Con_i, при которой в $\mathcal{E}_1$ выполняется условие $(Con_1 \& \ldots \& Con_k) \circ \sigma$ и последовательность операций $Act_1 \circ \sigma, \ldots, Act_m \circ \sigma$ преобразует $\mathcal{E}_1$ в $\mathcal{E}_2$. Выполнение условия при данной базисной подстановке в состоянии $\mathcal{E}$ понимается у нас несколько иначе чем в экспертных системах, а именно, когда зафиксирована базисная подстановка σ, то позитивное элементарное условие α выполняется в $\mathcal{E}$, если $\alpha \circ \sigma \in \mathcal{E}$; негативное элементарное условие $\neg\alpha$ выполняется в $\mathcal{E}$, если **никакой базисный частный случай** $\alpha \circ \sigma$ не принадлежит $\mathcal{E}$. Отношение $\vdash^\pi$, вообще говоря, является недетерминированным, поскольку условие π может выполняться при различных подстановках σ.

Отметим, что каждая продукция задается Хорновским предложением вида $q :- Con_1, \ldots, Con_k, Act_1, \ldots, Act_m$, в котором q - 0-местный интенсиональный предикат. Таким образом, классификация из определения 2 распространяется и на продукционные программы.

Определение 3 Продукционной динамической базой данных (продукционной ДБД) *мы назовем систему* $\mathcal{B} =< \mathcal{I}, \Phi, \equiv_i >$, *в которой программа* $\mathcal{I}$ *- конечное множество продукций,* Φ *- некоторая формула 1-го порядка в сигнатуре* $\mathbf{P}^e$, *описывающая* ограничения целостности (IC), *и* $\equiv_i$ *- отношение эквивалентности на* $\mathbf{B}^e$.

$\mathcal{B}$ *задает на множестве состояний отношение перехода* $\vdash_{\mathcal{I}} = \bigcup_{\pi \in \mathcal{I}} \vdash_\pi$, *и при этом* $\vdash_{\mathcal{I}}$ *и* Φ *согласованы с* $\equiv_i$.

Ключевым понятием предлагаемого подхода является понятие траектории с возмущениями, которое отражает интерактивное поведение ДБД во взаимодействии с ее средой.

Определение 4 *Для любых двух непересекающихся множеств* $\mathcal{D}^+, \mathcal{D}^-$ *базисных фактов определим* $(\mathcal{D}^+, \mathcal{D}^-)$-возмущение *как отношение* $\mathcal{E}_1 \stackrel{\mathcal{D}^+, \mathcal{D}^-}{\longrightarrow} \mathcal{E}_2$ *на состояниях БД такое, что* $\mathcal{E}_2 = (\mathcal{E}_1 \cup \mathcal{D}^+) \setminus \mathcal{D}^-$.

Пусть $\mathcal{B} =< \mathcal{I}, \Phi, \equiv_i >$ *- продукционная ДБД. Мы определим* траекторию $\mathcal{B}$ *как конечную или бесконечную последовательность вида*

$$\omega : \ \mathcal{E}_0 \vdash_{\mathcal{I}} \mathcal{E}_1^* \stackrel{\mathcal{D}_1^+, \mathcal{D}_1^-}{\longrightarrow} \mathcal{E}_1 \vdash_{\mathcal{I}} \mathcal{E}_2^* \stackrel{\mathcal{D}_2^+, \mathcal{D}_2^-}{\longrightarrow} \mathcal{E}_2 \ \ldots \ .$$

Мы называем такие траектории (начинающиеся с модификаций) M-траекториями. *Двойственные траектории, начинающиеся с возмущений, мы называем* B-траекториями. *Пусть* $\mu(\mathcal{E}, \mathcal{D}^+, \mathcal{D}^-)$ *- некоторое (полиномиально вычислимое) свойство троек конечных множеств фактов. Тогда* ω *является* $\mu-$ограниченной, *если для всех* $k \geq 1$ *выполнено условие* $\mu(\mathcal{E}_k^*, \mathcal{D}_k^+, \mathcal{D}_k^-)$. *M-траектория (B-траектория) является* стабильной *(соответственно,* гомеостатичной,*) если каждое ее состояние* $\mathcal{E}_i$, $i = 0, 1, 2, \ldots$ *удовлетворяет IC* Φ, *т.е.* $\mathcal{E}_i \models \Phi$.

Стабильность траектории означает, что все нарушения IC, вызванные модификациями ДБД, компенсируются на ней последующими возмущениями. Двойственное понятие гомеостатичности означает, что все нарушения IC, вызванные возмущениями, компенсируются последующими модификациями ДБД. Заметим, что всегда можно было бы компенсировать нарушения IC за счет достаточно больших возмущений или, в двойственном случае, за счет достаточно сложных

модификаций. Поэтому, чтобы получить адекватное определение устойчивого поведения, мы и вводим ограничение μ на компенсирующие или компенсируемые возмущения. В предыдущих работах [2, 3, 4] мы рассматривали важные специальные виды условий μ. Именно, пусть $\delta =< \mathcal{D}^+, \mathcal{D}^- >$ - пара конечных множеств фактов. Положим

$$\{\delta\}(d^+, d^-) \ \rightleftharpoons \ d^+ \Leftarrow \mathcal{D}^+ \ \& \ d^- \Leftarrow \mathcal{D}^-.$$

Другие виды ограничений, которые мы используем в примерах, связаны с ограничениями мощности возмущений:

$$\{\delta\}^m(d^+, d^-) \rightleftharpoons \{\delta\}(d^+, d^-) \ \& \ (|d^+| + |d^-|) \leq m.$$

Для этих важных видов ограничений μ имеются и соответствующие типы ограниченных траекторий - $\{\delta\}-$ограниченные и $\{\delta\}^m-$ограниченные.

Понятия $\mu-$устойчивости траекторий естественным образом приводят к различным понятиям $\mu-$устойчивости поведения ДБД в фиксированном состоянии БД. Ясно, что множество всех $\mu-$ограниченных траекторий, начинающихся в данном состоянии $\mathcal{E}$, образует дерево T, вообще говоря, бесконечной глубины. Под $\forall\exists-$поддеревом дерева T понимается, как обычно, такое его поддерево, в котором каждая внутренняя вершина четной глубины имеет всех тех же сыновей, что и в дереве T, а каждая внутренняя вершина нечетной глубины имеет в точности одного сына.

Определение 5 *Пусть даны некоторая ДБД $\mathcal{B}$ и свойство $\mu(\mathcal{E}, \mathcal{D}^+, \mathcal{D}^-)$. $\mathcal{B}$ является $\mu-\exists\exists$-стабильной в состоянии БД $\mathcal{E}$, если у нее есть бесконечная μ-ограниченная стабильная траектория, начинающаяся в $\mathcal{E}$.*

$\mathcal{B}$ является $\mu-\forall\exists$-гомеостатичной в состоянии БД $\mathcal{E}$, если в дереве всех ее μ-ограниченных B-траекторий, начинающихся в $\mathcal{E}$, имеется $\forall\exists$-поддерево, в котором все ветви являются бесконечными гомеостатичными траекториями.

*Состояние БД $\mathcal{E}$ назовем $\mu-$*перспективным* *для $\mathcal{B}$, если существует конечная μ-ограниченная траектория, начинающаяся в $\mathcal{E}$ и заканчивающаяся в некотором состоянии БД, удовлетворяющем IC* Ф. *Мы назовем состояние $\mathcal{E}$* перспективным*, если оно является $\{\delta\}$-перспективным для пустого δ.*

Стабильность и гомеостатичность характеризуют поведение системы в допустимых состояниях, а перспективность позволяет классифицировать недопустимые состояния на те, из которых можно попасть в допустимые, за счет подходящих модификаций ДБД, и те, из которых нельзя.

Пример 1. Рассмотрим утрированно примитивную ДДБ $\mathcal{B}_1$, описывающую состояние здоровья с помощью трех пропозициональных экстенсиональных предикатов *healthy*, *ill* и *medicine* (последний выражает наличие необходимого лекарства). Продукционная программа состоит из двух продукций:

$upd_1 :$ $ill\&medicine \ \Rightarrow \ delete(ill), delete(medicine), insert(healthy)$.

$upd_2 :$ $healthy \ \Rightarrow \ change(healthy, ill)$.

Ограничение целостности ϕ_1 описывает допустимые состояния:

$healthy \ \vee (ill \ \& \ medicine)$.

При $\delta = (\{medicine\}, \emptyset)$ эта продукционная ДБД является $\{\delta\}-\exists\exists$-стабильной в любом состоянии БД, удовлетворяющем IC. Действительно, если состояние БД содержит *healthy*, то можно применить продукцию upd_2 . Это может нарушить IC, если в состоянии нет факта *medicine* . Однако, IC восстанавливается при непустом возмущении. Если же в состоянии имеются *ill* и *medicine*, то применима первая продукция upd_1 и возникнет состояние, содержащее *healthy* . Отметим также, что ни одно из состояний БД, в котором нарушено IC, не является перспективным для $\mathcal{B}_1$. Однако, если заменить ограничение целостности ϕ_1 на

следующее ϕ_2 :

$healthy \vee \neg ill$,

то хотя состояние $\{ill, medicine\}$ и не удовлетворяет новому IC ϕ_2, оно является перспективным.

Пример 2. Расширим предыдущий пример, допуская две различные болезни ds_1, ds_2 и соответствующие им лекарства mdc_1, mdc_2. Пусть также в результате лечения этих болезней возникает иммунитет imm_1, imm_2. Новая ДБД $\mathcal{B}_2$ имеет следующую программу:

$upd_{1i}: ds_i \& mdc_i \Rightarrow delete(ds_i), delete(mdc_i), insert(imm_i). (i = 1, 2),$

$upd_2: \neg imm_1 \& \neg imm_2 \& \neg ds_1 \& \neg ds_2 \Rightarrow insert(ds_1), insert(ds_2).$

$upd_{3i}: imm_i \Rightarrow delete(imm_i). (i = 1, 2)$

и ограничение целостности IC:

$(\neg ds_1 \vee mdc_1) \ \& \ (\neg ds_2 \vee mdc_2)$.

ДБД $\mathcal{B}_2$ является $\{\delta\} - \exists\exists$-стабильной для $\delta = (\{mdc_1, mdc_2\}, \emptyset)$ во всех состояниях, удовлетворяющих IC. Это достаточно очевидно, потому что в каждом состоянии БД, удовлетворяющем IC, применима хотя бы одна продукция и, кроме того, максимальное возмущение переводит любое состояние в состояние, удовлетворяющее IC. Однако, для $\delta_1 = (\{mdc_1\}, \emptyset)$ $\mathcal{B}_2$ не является $\{\delta_1\} - \exists\exists$-стабильной ни в каком состоянии. Причина этого в том, что всякая бесконечная траектория должна содержать применение продукции upd_2, которая приводит к состоянию, включающему обе болезни. Тогда для восстановления IC необходим факт mdc_2 , которого нет в δ_1.

Пример 3. Еще раз изменим этот пример. Пусть заражение одной из двух различных болезней является возмущением внешней среды $\delta = (\{ds_1, ds_2\}, \emptyset)$, а соответствующие болезням лекарства mdc_1, mdc_2 должны закупаться при отсутствии иммунитета, который, как и в примере 2 возникает в результате лечения болезней. ДБД $\mathcal{B}_3$ имеет следующую программу:

$upd_{1i}: ds_i \& mdc_i \Rightarrow delete(ds_i), delete(mdc_i), insert(imm_i). (i = 1, 2),$

$upd_{2i}: ds_i \& imm_i \Rightarrow delete(ds_i), delete(imm_i), insert(mdc_i). (i = 1, 2)$

$upd_3: \neg ds_1 \& \neg ds_2 \Rightarrow$.

и ограничение целостности IC:

$(\neg ds_1 \& \neg ds_2)$.

Продукция upd_3 с пустой правой частью означает, что при отсутствии болезней никакие действия не предпринимаются. Нетрудно проверить, что ДБД $\mathcal{B}_3$ является $\{\delta\}^1 - \forall\exists$-гомеостатичной в каждом из состояний $\{mdc_1, mdc_2\}$, $\{imm_1, mdc_2\}$, $\{mdc_1, imm_2\}$, $\{imm_1, imm_2\}$, т.е. при условии, что лишь один факт из δ может попадать в состояние при каждом возмущении. Но ни в одном состоянии она не является $\{\delta\} - \forall\exists$-гомеостатичной, поскольку ее правила не позволяют справляться с двумя болезнями одновременно.

Рассмотрим следующую естественную классификацию продукционных ДБД по типам используемых продукций. Обозначим через $PROD$ множество всех продукционных ДБД. К сожалению, в этом классе многие интересующие нас задачи анализа поведения систем неразрешимы. Поэтому мы выделяем внутри него несколько подклассов:

- $PROF$ - подкласс всех ДБД из $PROD$ с плоскими программами;
- $PROG$ - подкласс всех ДБД из $PROD$ с базисными программами;
- $PROG^-$ - подкласс всех ДБД из $PROG$ с расширяющими программами;
- $PROG^+$ - подкласс всех ДБД из $PROG^-$ с позитивными программами (т.е. продукции в программах из $PROG^+$ не содержат ни отрицаний, ни удалений $delete/1$).

ДБД $\mathcal{B}_1$ в примере 1 принадлежит классу $PROG^-$, а $\mathcal{B}_2$ и $\mathcal{B}_3$ в примерах 2 и 3 входят в $PROG$. Во всех указанных классах можно найти много важных приложений. Например, так называемые продукционные экспертные системы попадают в класс $PROF$, а некоторые из них даже в класс $PROG^-$.

Что же касается ограничений целостности, то мы выберем в качестве IC некоторые подклассы формул 1-го порядка, которые с одной стороны покрывают многие классы приложений, а с другой - являются относительно просто проверяемыми. Наболее сильное ограничение на IC, которое мы накладываем - это *базисность,* т.е. отсутствие переменных, плюс *монотонность*

(отсутствие отрицаний) формул. В нашей классификации это ограничение называется $IC0-$*критерием.* Следующий рассматриваемый нами класс состоит из базисных формул, это ограничение мы называем $IC1-$*критерием.* Еще более широким является класс IC, задаваемых $\exists-$формулами. Мы будем говорить, что ограничения из этого класса удовлетворяют $IC2-$*критерию.*

Приведем установленные в работах [3, 4, 5] факты, характеризующие разрешимость и вычислительную сложность проблем распознавания различных видов устойчивости.

- Наиболее узким классом рассматривемых систем является класс продукционных ДБД из $PROG^+$ с $IC0-$критериями. Задача перспективности для этого случая эквивалентна хорошо изученным проблемам вычисления базиса функциональных зависимостей в реляционных БД, синтеза программ в простых моделях Тыугу и т.п. Она может быть решена за линейное время ([6]). Та же проблема с $IC1-$критериями $NP-$полна.
- В классе $PROG^-$ даже с $IC0-$критериями эта проблема $PSPACE-$полна.
- Еще более сложной (хотя и разрешимой) проблема перспективности является в классе $PROF$. В нем она является экспоненциальной по памяти уже с $IC0-$критериями.
- В классе $PROD$ всех продукционных ДБД проблема перспективности неразрешима.
- Хоть это может показаться странным, но сложность проблемы $\{\delta\}-\exists\exists-$стабильности в перечисленных классах ДБД та же, что и у проблемы перспективности [3, 4].
- Что касается проблемы $\{\delta\}-\forall\exists-$гомеостатичности, то она в целом сложнее. Так, в классе ДБД $PROG^+$ с $IC0-$критериями эта проблема имеет полиномиальную временную сложность. Но уже для немонотонных $IC1-$ критериев ее сложность резко возрастает до экспоненциального времени.
- Для решения проблемы $\{\delta\}-\forall\exists-$гомеостатичности в классе ДБД $PROF$ понадобится уже алгоритм с временем работы, оцениваемым двойной экспонентой от длины входа.
- Разумеется, в классе $PROD$ эта проблема неразрешима.

4 Дедуктивные динамические БД

Рассмотренные в предыдущем разделе продукции являются весьма частным случаем предложений логических программ. Ограниченность таких предложений состоит в отсутствии рекурсии, позволяющей описывать сложные преобразования состояний за один шаг работы системы. В этом разделе мы рассматриваем более общие дедуктивные динамические базы данных, в которых вместо продукционных допускаются произвольные логические программы в сигнатуре $\mathbf{P}^e \cup \mathbf{P}^i$. Предложения таких программ имеют вид

$$p(\bar{t}) :- \alpha_1, \ldots, \alpha_n,$$

где $p \in \mathbf{P}^i$, и при этом, если $\alpha_i = \neg q(\bar{t})$, $\alpha_i = insert(q(\bar{t}))$ или $\alpha_i = delete(q(\bar{t}))$, то $q \in \mathbf{P}^e$. Предлагаемое общее определение дедуктивных ДБД соответствуют схеме, описанной в обзоре [7]. Они включают интенсиональную логическую программу с модификациями, заранее определенный набор целей, реализующих преобразования состояний БД, ограничения целостности, заданные логической формулой, и некоторое отношение эквивалентности на данных.

Определение 6 Дедуктивная динамическая база данных(дедуктивная ДБД) - *это система*

$$\mathcal{B} =< \mathcal{I} \cup \{ :- G_1, \ldots, :- G_n \}, \Phi, \equiv_i >$$

где:

- $\mathcal{I}$ - *логическая программа с модификациями,*

- *все цели* G_i, $i = 1, \ldots, n$, *являются либо базисными, либо стационарными (т.е. не вызывающими изменений состояний),*

- *с каждой базисной целью* $:- G_i$ *связывается недетерминированный оператор на состояниях БД (модификация)* $\overset{G_i}{\vdash_{\mathcal{I}}}$ *(или просто* $\overset{G_i}{\vdash}$, *если* $\mathcal{I}$ *подразумевается) определяемый естественным образом:* $\mathcal{E} \overset{G_i}{\vdash_{\mathcal{I}}} \mathcal{E}'$ *тогда и только тогда, когда существует успешное опровержение ([8, 13])* $\mathcal{I} \cup \mathcal{E} \cup \{ :- G_i\} \Rightarrow, \Box$ *завершающееся в состоянии* $\mathcal{E}'$,

- Φ - *некоторая формула 1-го порядка в экстенсиональной сигнатуре, задающая* ограничения целостности (IC),

- $\equiv_i$ - *эквивалентность на* $\mathbf{B}^e$; *все отношения* $\overset{G_i}{\vdash_{\mathcal{I}}}$ *и формула* Φ *согласованы с* $\equiv_i$.

Базисные цели назовем модификациями, *а стационарные* - запросами.

Понятие траектории с возмущениями, отражающее интерактивное поведение дедуктивной ДБД во взаимодействии с ее средой, определяется здесь так же, как и для продукционных ДБД. При этом однако значительно более мощными становятся модификации. Там где в продукционной ДБД состояние БД $\mathcal{E}_i$ преобразуется в состояние $\mathcal{E}_{i+1}$ применением некоторой продукции, там в дедуктивной ДБД происходит успешное вычисление логической программы $\mathcal{I}$ с состоянием БД $\mathcal{E}_i$ и с одной из целей $\{ :- G_i\}$, приводящее к состоянию $\mathcal{E}_{i+1}$. При этом свойства устойчивого поведения дедуктивных ДБД: $\mu-\exists\exists$- стабильность, $\mu-\forall\exists$-гомеостатичность, $\mu-$перспективность и др, определяются точно так же, как и для продукционных ДБД.

Класс произвольных логических программ по вычислительной силе эквивалентен классу машин Тьюринга. Поэтому большинство естественных алгоритмических проблем, связанных с преобразованиями состояний БД в нем неразрешимы. Тем не менее, можно наложить на логические программы естественные ограничения, часто выполняющиеся в приложениях и обеспечивающие разрешимость ряда проблем. Одим из наиболее реалистических ограничений этого рода является свойство стратифицируемости, аналогичное в некотором смысле свойству, описанному в [1].

Определение 7 *Пусть* $\mathcal{P}$ - *логическая программа. Скажем, что предикат* p ссылается на *предикат* q, *если в* $\mathcal{P}$ *есть предложение, определяющее* p, *с вызовом* q *в его теле. Пусть отношение* "зависит от" *является рефлексивным и транзитивным замыканием отношения "ссылается на". Максимальные сильно связные компоненты графа отношения "зависит от"назовем* кликами. $\mathcal{P}$ *называется* динамически стратифицированной (Д-стратифицированной), *если для всякого ее предложения*

$$p(\bar{t}) :- p_1(\bar{t_1}), \ldots, p_i(\bar{t_i}), q(\bar{u}), p_{i+1}(\bar{t}_{i+1}), \ldots, p_r(\bar{t_r}),$$

в котором q *входит в клику* p, *все предикаты* $p_1, \ldots, p_i, p_{i+1}, \ldots, p_r$ *являются* стационарными, *т.е. не зависят от элементарных модификаций.*

Смысл этого определения состоит в том, что модификации БД могут производиться лишь на тех шагах, на которых меняется клика вызываемого предиката. Мы рассматриваем два класса дедуктивных ДБД с Д-стратифицированной рекурсией:

- GDS - класс всех ДБД с Д-стратифицированными и базисными интенсиональными частями;

- FDS - класс всех ДБД с Д-стратифицированными и плоскими интенсиональными частями.

Из этих определений непосредственно следует, что $PROG \subset GDS$ и $PROF \subset FDS$.

Приведем пример дедуктивной Д-стратифицированной ДБД.

Пример 4. Рассмотрим завод по переработке мусора, на котором процесс переработки осуществляется циклами. На каждом цикле перерабатывается CA единиц мусора, что задается фактом $cycle\_amount(CA)$. Завод использует резервуар для собранного мусора; текущее количество мусора в нем задается фактом $reserve(Res)$. Каждый цикл переработки приносит заводу доход, размер которого указывает факт $gain(R^+)$. Текущие средства предприятия

фиксируются фактом $resources(Rr)$. Обычно мусор собирается в количестве $Coll$ и доставляется на завод бесплатно. Это отражается внешним возмущением $D^+ = \{collected(Coll)\}$. Однако при перебоях в завозе мусора, и в ситуации, когда в резервуаре опасно мало запаса (т.е. не более чем $2 * CA$), завод вынужден собирать CA единиц мусора своими силами, чтобы не прерывать производственный цикл. На это он вынужден тратить сумму, описываемую фактом $spend\_rate(R^-)$. В каждый момент запас мусора Res в резервуаре должен удовлетворять условию: $\neg(Res \leq CA \ \vee \ Res > 2 * CA) \ \vee \ (Rr \leq R^-)$. Это условие задает ограничения целостности. Само же производство описывается следующей логической программой $\mathcal{I}$:

$utilize$:–
$amount(Total, CA)$,
$Total > 2 * CA$,
$utilize\_collected$.
$utilize$:–
$amount(Total, CA)$,
$Total \leq 2 * CA$,
$delete(collected(X))$,
$change(reserve(Res), reserve(Total))$,
$collect\_utilize$.
$utilize\_collected$:–
$amount(Total, CA)$,
$divide(Total, CA, Quotient, Rest)$,
$delete(collected(X))$,
$(Rest > 0, Supply\ is\ CA + Rest, Q\ is\ (Quotient - 1)$;
$Rest == 0, Supply\ is\ 2 * CA, Q\ is\ (Quotient - 2))$,
$change(reserve(Res), reserve(Supply))$, % Q cycles.
$gain(R^+)$,
$change(resources(Rr), resources(Rr + Q * R^+))$.
$collect\_utilize$:–
$\neg collected(X)$,
$reserve(Res)$,
$cycle\_amount(CA)$,
$CA < Res, Res \leq 2 * CA$,
$resources(Rr)$,
$spend\_rate(R^-)$,
$Rr > R^-$,
$save\_rate(R^+)$,
$change(resources(Rr), resources(Rr - R^- + R^+))$.

$amount/2$ - это следующий стационарный предикат, вычисляющий общее количество имеющегося мусора:

$amount(Total, CA)$:–
$reserve(Res)$,
$cycle\_amount(CA)$,
$Res > CA$,
$(collected(Coll), TotalisRes + Coll$;
$\neg collected(X), Total = Res)$.

$divide(X, Y, Q, R)$ - это встроенный арифметический предикат, который по двум целочисленным входам X и Y выдает в качестве результата частное Q и остаток R от деления X на Y. Ограничения целостности IC задаются формулой:

$\Phi = (reserve(Res)\ \&\ cycle\_amount(CA)\ \&\ resources(Rr)\ \ \&\ spend\_rate(R^-)) \Rightarrow$
$(\ CA < Res\ \&\ Res \leq 2 * CA\ \&\ Rr > R^-\ ).$

В программе $\mathcal{I}$ целевая модификация $utilize$ зависит от двух динамических предикатов $collect\_utilize$ и $utilize\_collected$, каждый из которых зависит лишь от стационарных предикатов. Таким образом, $\mathcal{I}$ является Д-стратифицированной, а ДБД $\mathcal{B} = < \mathcal{I} \cup \{utilize\}, \Phi, => $ принадлежит классу FDS. Нетрудно проверить, что если в начальном допустимом состоянии $\mathcal{E}$ $R^- < R^+$, то $\mathcal{B}$ $\{\delta\} - \forall\exists$-гомеостатична в $\mathcal{E}$ при $\delta =< \{collected(Coll)\}, \emptyset >$ (мы предполагаем, что множество возможных значений $Coll$ конечно). Действительно, в этом случае, даже если на завод не поступил извне мусор (факт $collected(Coll)$), внутренних ресурсов хватит на выполнение очередного цикла и объем ресурсов после него не уменьшится.

Один из наиболее важных результатов работ [3, 4, 5] связывает между собой верхние оценки сложности распознавания свойств устойчивого поведения рекурсивных Д-стратифицированных ДБД и продукционных ДБД. Несмотря на то, что классы ДБД GDS и FDS расширяют соответствующие продукционные классы $PROG$ и $PROF$, верхние оценки сложности рассматриваемых вариантов задач перспективности, стабильности и гомеостатичности для них совпадают с оценками для продукционных классов, иначе говоря, для их решения имеются алгоритмы той же сложности, что и для аналогичных классов продукционных ДБД. В частности:

- Задачи проверки перспективности и $\{\delta\} - \exists\exists$-стабильности для класса GDS являются $PSPACE-$полными.
- Задача проверки $\{\delta\} - \forall\exists$-гомеостатичности для GDS требует времени, оцениваемого двойной экспонентой от длины входа.
- Проблемы перспективности и $\{\delta\}-\exists\exists$-стабильности являются в классе FDS экспоненциальными по памяти уже с $IC0-$критериями.
- Для решения проблемы $\{\delta\} - \forall\exists-$гомеостатичности в классе ДБД FDS требуется алгоритм с временем работы, оцениваемым двойной экспонентой от длины входа.

Как мы уже отмечали, все перечисленные здесь результаты работ [2, 3, 4, 5] были получены для $\{\delta\}-$ограниченных возмущений. Однако анализ построенных алгоритмов показывает, что они справедливы и для значительно более широкого класса *полиномиальных* $\{\delta\}-$ограниченных возмущений.

Определение 8 *Пусть* $\delta =< \mathcal{D}^+, \mathcal{D}^- >$ *- пара конечных множеств фактов. Вычислимое в полиномиальное время отношение* $\mu(\mathcal{E}, d^+, d^-)$ *является полиномиальным* $\{\delta\}-$*ограничением, если для любой тройки* $\mathcal{E}, d^+, d^-$ *конечных множеств фактов из* $\mathbf{B}^e$

$$\mu(\mathcal{E}, d^+, d^-)\ \Rightarrow\ d^+ \Lleftarrow \mathcal{D}^+\ \&\ d^- \Lleftarrow \mathcal{D}^-.$$

Имеет место следующая общая

Теорема. *Проблемы проверки свойств* $\mu - \exists\exists$*-стабильности и* $\mu - \forall\exists$*-гомеостатичности для полиномиальных* $\{\delta\}-$*ограничений* μ *имеют для всех рассматриваемых классов ДБД те же оценки сложности, что и соответствующие проблемы* $\{\delta\} - \exists\exists$*-стабильности и* $\{\delta\} - \forall\exists$*-гомеостатичности.*

5 Инструментальная поддержка

Приведенные выше факты, говорящие о переборном характере проблем $\exists\exists-$стабильности и $\forall\exists$-гомеостатичности, могут внушить мысль, что не может быть и речи об инструментальной поддержке анализа свойств устойчивости поведения дискретных динамических систем. Однако

это не так. Разумным компромиссом является интерактивное рабочее место специалиста в прикладной области, выполняющее анализ устойчивости состояния на основе функций локальных преобразований состояний или трассировки состояний на заданную глубину. Такие функции могут быть реализованы за практически приемлемое полиномиальное время. Достаточно полный набор подобных функций может обеспечить успешное использование системы для широкого круга инженерных задач. Обогащение инструментальной системы функциями глобального автономного анализа устойчивости поведения для классов систем, в которых они имеют приемлемую сложность, может значительно расширить ее возможности. Приведем примерный перечень возможных функций инструментальной системы поддержки анализа поведения дискретных динамических систем. Будем считать, что ДБД представляется в этой системе в виде двух центральных компонент: базы данных (БД) для представления экстенсиональных состояний и базы знаний (БЗ), реализующей интенсиональные логические программы.

Инструментальные функции:

1. Редактирование схемы БД.
2. Редактирование состояния, хранимого в БД.
3. Редактирование ограничений целостности состояний, автоматический анализ их сложности (по отношению к используемой классификации).
4. Проверка ограничений целостности на состояниях, интеллектульная диагностика причин их нарушения.
5. Для отношений эквивалентности, выразимых в простом языке, редактирование определений отношений, проверка эквивалентности атомов и состояний, проверка согласованности модификаций ДБД и ограничений целостности с эквивалентностями.
6. Редактирование описаний возмущений внешней среды и $\mu-$ограничений на возмущения, выразимых в простом языке.
7. Выбор возмущения, проверка его $\mu-$ограниченности и определение результата его применения к состоянию.
8. (Интерактивное) перечисление допустимых (недопустимых) состояний, возникающих из данного состояния в результате $\mu-$ограниченного возмущения.
9. Проверка существования допустимого (недопустимого) состояния, возникающего из данного состояния в результате $\mu-$ограниченного возмущения.
10. Редактирование БЗ.
11. Определение класса БЗ в сооветствии с используемой классификацией.
12. Анализ применимости модификаций к данному состоянию.
13. Определение результата применения продукции (для продукционной ДБД) и проверка его допустимости.
14. Для произвольной ДБД (интерактивное) перечисление допустимых (недопустимых) результатов применения модификации к данному состоянию.
15. Проверка достижимости из данного состояния некоторого допустимого (недопустимого) состояния в результате одной модификации.
16. Для заданных состояния и ограничения μ проверка существования допустимого (недопустимого) состояния, достижимого в результате одной модификации после некоторого (всех) $\mu-$ограниченного возмущения.
17. Для заданных состояния и ограничения μ проверка существования допустимого (недопустимого) состояния, достижимого в результате $\mu-$ограниченного возмущения после одной (всех) модификации.
18. Для каждого типа устойчивого поведения интерактивная трассировка траектории (возможно, в режиме интеллектуальной подсказки очередного шага и с возможностью возвратов); анализ "тупиковых"состояний.
19. Выполнение всех перечисленных локальных функций в режиме "что-если"(т.е. с внесением временных изменений периода исполнения в БД и в БЗ).
20. Для различных типов устойчивого поведения построение (перечисление, проверка сущест-

вования) деревьев траекторий соответствующего вида заданной глубины k.

21. Для различных видов стабильного, гомеостатического и перспективного поведения автономная проверка выполнения этих свойств в данном состоянии (возможно при заданных ограничениях на время проверки).

22. Проверка независимости устойчивых траекторий (деревьев траекторий) заданной длины (глубины) от фиксированных "небольших"изменений исходного состояния. Определение "точек расхождения"при отрицательном исходе проверки.

23. Синтез устойчивых в разных смыслах траекторий и стратегий поведения вдоль этих траекторий.

Даже сам перечень инструментальных функций показывает, что при наличии удобного интерфейса предоставляющая их система может оказаться эффективным средством анализа поведения дискретных динамических систем.

6 Заключение

В работе предложена новая модель интерактивного поведения дедуктивных баз данных и определены некоторые характеристики его устойчивости по отношению к возмущениям внешней среды. Даже весьма упрощенные примеры, приведенные нами, показывают, что разнообразные динамические системы, взаимодействующие со внешней средой, могут моделироваться ДБД, а свойства устойчивости их поведения можно формулировать и исследовать в предложенных терминах. Полученные результаты показывают, что обсуждаемые в работе свойства устойчивого поведения - $\exists\exists$-стабильность и $\forall\exists-$гомеостатичность оказываются разрешимыми в ряде классов ДБД, интересных для приложений. Следует отметить, что соответствующие алгоритмы в общем случае либо работают в полиномиальное время при весьма строгих, но достаточно разумных ограничениях, либо являются простыми алгоритмами вида "породи-проверь", требующими экспоненциального времени или памяти. Для конкретных классов задач может оказаться, что такие недетерминированные алгоритмы могут быть реализованы и более эффективно. К тому же, и для продукционных, и для Д-стратифицированных ДБД, модификации, ограниченные возмущения и ограниченный в глубину анализ устойчивости реализуемы полиномиальными алгоритмами. Поэтому мы предполагаем, что, используя предложенный подход, можно разработать интерактивное программное окружение, предоставляющее широкий спектр функций для моделирования и анализа поведения сложных динамических систем.

Благодарности

Авторы очень признательны за полезное обсуждение этой работы Марсу Котдусовичу Валиеву и участникам семинаров в университете Париж-12, Московском государственном университете и Тверском государственном университете.

Список литературы

[1] Apt, K.R., Blair, H. and Walker A., *Towards a theory of declarative knowledge.* in: J. Minker (ed.) *Foundations of deductive databases and logic programming.* Morgan Kaufman Pub., Los Altos, 89-148, 1988.

[2] Дехтярь М.И., Диковский А.Я., *Динамические дедуктивные базы данных.* Известия РАН, Техническая кибернетика N 5, 1994, 55-66.

[3] Dekhtyar, M.I., Dikovsky, A.Ja., *Dynamic Deductive Data Bases with Steady Behavior.* In *Proc. of the 12th International Conf. on Logic Programming,* (Ed. L. Sterling), The MIT Press, 183-197, 1995.

[4] Dekhtyar, M.I., Dikovsky, A.Ja., *Properties of Steady Behavior of Deductive Data Bases .Part I. $\exists\exists-$stability and Promise.* Techn. rept. 95-07, Universite Paris XII, 1995.

[5] Dekhtyar, M.I., Dikovsky, A.Ja., *Properties of Steady Behavior of Deductive Data Bases . Part II. Stability and Homeostaticity.* Techn. rept. , Universite Paris XII, 1996.

[6] Dikovsky, A.Ja., *Linear time solutions to the problems related to automatic synthesis of acyclic programw.* Programming, 3, 1985.

[7] Gallaire, H., Minker, J., Nicolas, J.-M., *Logic and databases: a deductive approach.* ACM Computing Surveys, vol. 16, n.2, 153-185, 1984.

[8] Lloyd, J., *Foundation of logic programming.* Springer Verlag, 1987.

[9] Manchanda, S., Warren, D.S., *A logic-based language for database updates.* In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming,* Morgan-Kaufmann, Los Altos, CA, 363-394, 1988.

[10] Naqvi, S., Krishnamurthy, R., *Database updates in logic programming.* In *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 251-262, 1988.

[11] Naqvi, S., Tsur, S., *A Logical Language for Data and Knowledge Bases.* ComputerScience Press, New York, 1989.

[12] Sadri, F., Kowalski, R., *A theorem proving approach to database integrity.* In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming,* Morgan-Kaufmann, Los Altos, CA, 313-362, 1988.

[13] Хоггер К., *Введение в логическое программирование.* М., Мир, 1988.

[14] С.Чери, Г.Готлоб, Л.Танка, *Логическое программирование и базы данных.* М.: Мир, 1992.